
Introducción a Ruby on Rails

— Controllers y Views —

- Cómo generamos contenido dinámico
- Cómo generamos controllers
- Cómo generamos acciones
- Y qué es ERB

Generando un Controller

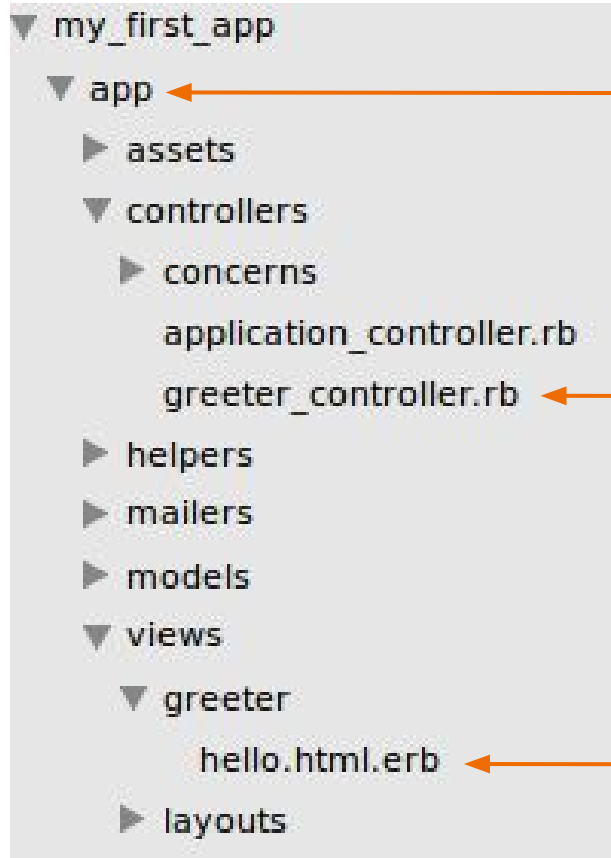
- Los controllers contienen **acciones** (métodos ruby) y orquestan los **request**.
- Rails puede generar **rápidamente** un controller con **0 o más acciones** y sus **vistas asociadas**.
- `rails generate controller controller_name [action1, action2, action3]`

Generando un controller

```
my_first_app git:(master) X rails g controller greeter hello
create  app/controllers/greeter_controller.rb
route  get "greeter/hello"
erb
create  app/views/greeter
create  app/views/greeter/hello.html.erb
test_unit
create  test/controllers/greeter_controller_test.rb
helper
create  app/helpers/greeter_helper.rb
test_unit
create  test/helpers/greeter_helper_test.rb
assets
coffee
create  app/assets/javascripts/greeter.js.coffee
scss
create  app/assets/stylesheets/greeter.css.scss
```

→ **controller**

→ **view**



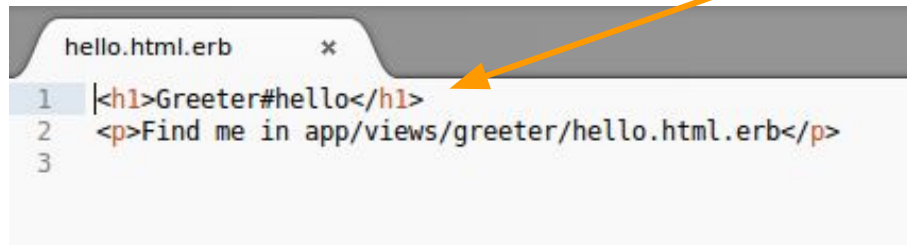
en el directorio app pasaremos la mayor parte del tiempo

El controller que creamos recién!

También se creó un view

Cómo es un view?

No se escriben los tags comunes de cualquier archivo html.. DOCTYPE, head, body.. etc.



```
hello.html.erb x
1 |<h1>Greeter#hello</h1>
2 <p>Find me in app/views/greeter/hello.html.erb</p>
3
```



Greeter#hello

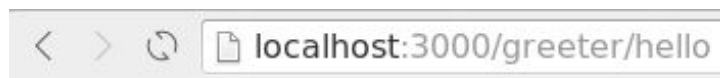
Find me in app/views/greeter/hello.html.erb

ERB (Embedded Ruby)

- El archivo view fue generado por rails y **luce como un archivo html**, pero tiene una **extensión .erb**
- ERB es una **librería de templating** (parecida a JSP, por ejemplo) que nos permite **embeber código Ruby dentro del HTML**.
- Existen solo dos patrones de etiquetas para aprender: (tag pattern)
 - **<% ... código ruby...%>** - Evalúa código ruby
 - **<%= ... código ruby... %>** - Imprime código ruby

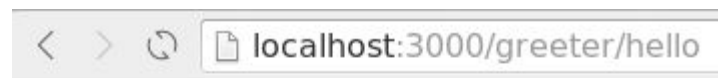
El nuevo hello.html.erb

```
hello.html.erb  ✕  
1  <%= random_names = ["vanessa", "iván", "eduardo"] %>  
2  <h1>Greetings, <%= random_names.sample %></h1>  
3  <p>La hora es: <%= Time.now %></p>
```



Greetings, iván

La hora es: 2016-10-30 21:07:48 -0300



Greetings, eduardo

La hora es: 2016-10-30 21:09:01 -0300

Cómo luce un controller.rb?

```
greeter_controller.rb ✕  
1  class GreeterController < ApplicationController  
2    def hello  
3    end  
4  end
```

- La acción **hello** es un método ruby regular (vacío en este caso)
- Qué pasa si queremos agregar una acción **goodbye** al **greeter** controller y también agregar un **goodbye.html.erb** al directorio **app/views/greeter**?

Creemos nuestra acción goodbye!

```
greeter_controller.rb x goodbye.html.erb x
1 class GreeterController < ApplicationController
2
3   def hello
4   end
5
6   def goodbye
7   end
8
9 end
```

Problemas!

```
▼ my_first_app
  ▼ app
    ► assets
    ▼ controllers
      ► concerns
        application_controller.rb
        greeter_controller.rb
      ► helpers
      ► mailers
      ► models
    ▼ views
      ▼ greeter
        goodbye.html.erb
        hello.html.erb
      ▼ layouts
        application.html.erb
```

Agregando la acción goodbye

< > ↺ localhost:3000/greeter/goodbye

Routing Error

No route matches [GET] "/greeter/goodbye"

Rails.root: /home/vanessa/git/capacitaciones/ruby-on-rails-intro/modulo-3/my_first_app

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

Routes

Routes match in priority from top to bottom

Helper	HTTP Verb	Path	Controller#Action
<u>Path / Url</u>		Path Match	
greeter_hello_path	GET	/greeter/hello(..:format)	greeter#hello

Entonces..

- Los **controllers** contienen **acciones** (las acciones son métodos)
- ERB permite tanto la **evaluación** (<%%>) de código ruby como el **despliegue** (<%= %>) del resultado de la información en la vista.

Routes..